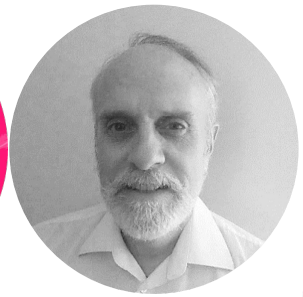
The Adapter Pattern



Gerald Britton

Pluralsight Author

@GeraldBritton www.linkedin.com/in/geraldbritton

OOP Principles

- Program towards abstractions, not implementations
- The ‘D’ in SOLID
- The ‘O’ in SOLID
- Open/Closed principle

Adapters in Real Life

Wall wart

Pipe adapter

Don’t try this at home!

Motivation

Print names and addresses

Customer object

Make it work with vendor objects

Vendor API is different

Customer: address property

Vendor: number and street properties

Make a new version of your program

Violates don’t repeat yourself (DRY)

Conditional logic

Demo

Start with original program

Prints customer names and addresses

Modify it to support vendors as well

Adapter

Classification: Structural

Converts interface of a class

Into another that clients expect

Lets classes work together

Can provide additional functionality

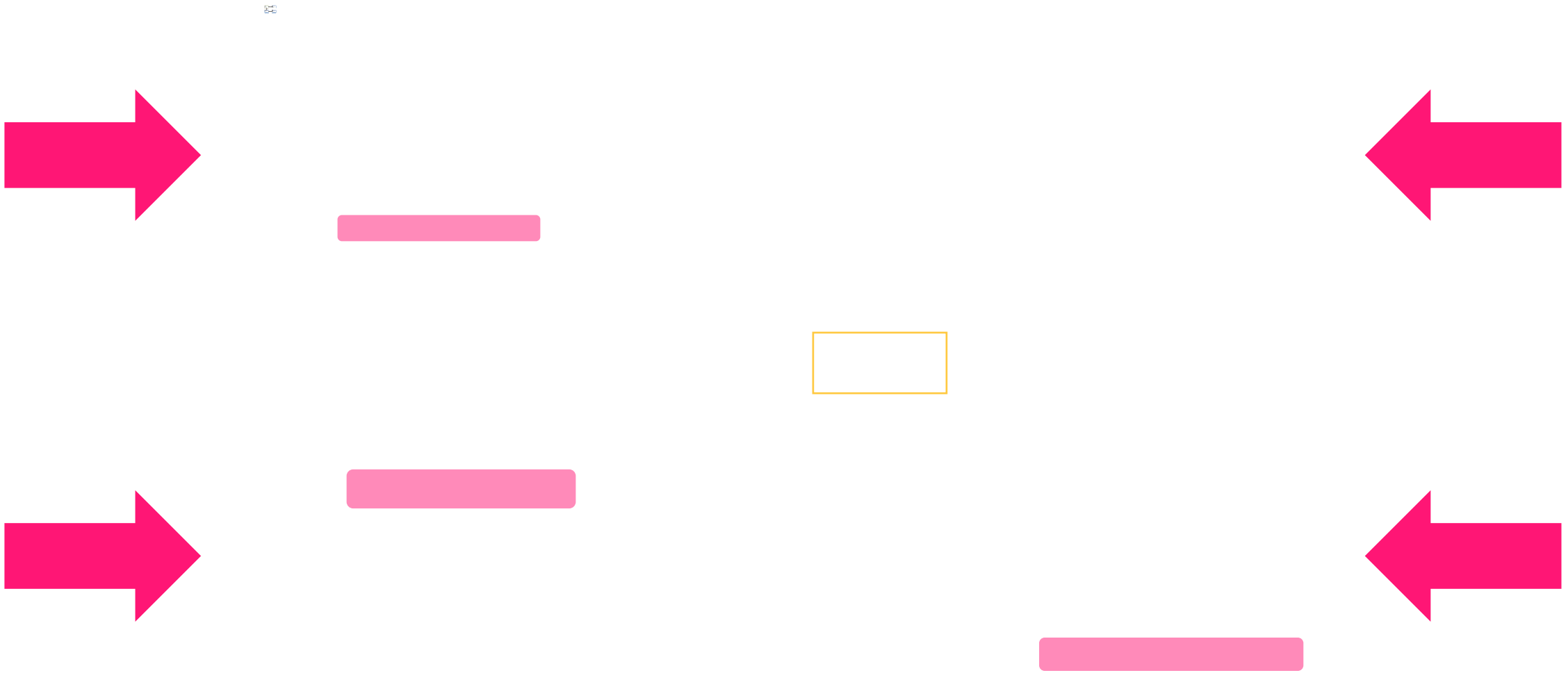
Two types of adapters

- Object adapters: Composition
- Class adapters: Inheritance

Favor composition over inheritance

Also known as the wrapper pattern

Object Adapter Structure



Demo

Implementing an object adapter

Class Adapter Structure

Demo

Implementing a class adapter

Pros and Cons

Object Adapter

Composition over inheritance

Delegate to the adaptee

Works with all adaptee subclasses

Class Adapter

Subclassing

Override adaptee methods

Committed to one adaptee subclass

Summary

Adapt an interface to the one you need

Create reusable code

New, unrelated, or unforeseen interfaces

Object Adapter: several subclasses

Class Adapter: One subclass

Which one should you use?

It depends!